Multimedia Tools and Applications (2022) 81:37119–37135
https://doi.org/10.1007/s11042-022-13542-3

1211: AIOT SUPPORT AND APPLICATIONS WITH MULTIMEDIA

Comparative analysis of CN2 rule induction with other classification algorithms for network security

Neeraj Kumar[1] · Upendra Kumar[1]

Received: 29 October 2020 / Revised: 18 August 2021 / Accepted: 30 May 2022/
Published online: 9 August 2022

Abstract

Network Intrusion Detection (NID) is an important domain of research and as the network traffic is growing enormously, new challenges are emerging forth in terms of quality of service and security thus a reliable and robust network intrusion detection system (NIDS) has become an inevitable requirement. Researchers have developed and implemented many rule-based and statistical-based techniques, as well as machine learning (ML) approaches such as K-Nearest Neighbour (K-NN), Decision Tree (DT), Random Forest (RF), Multi-Layer Perceptron (MLP) and Naive-Bayes for this pursuit. But, a method is still in need that entails certain required characteristics like low computational cost, high classification accuracy, able to deal efficiently with large datasets having redundant data and high dimensionality. In this paper, a novel method is proposed that includes all these characteristics. Further, the proposed method is based on the rule induction technique using separate and-conquer algorithm known as CN2 for a network intrusion detection system. The performance of the proposed CN2 rule induction algorithm is also compared experimentally with K-NN, DT, RF, MLP and Naive-Bayes classifiers for the Kddcup99 dataset and it has been observed that the performance of the CN2 rule induction algorithm is better in comparison to other classifiers considered in this experimentation.

Keywords Machine learning · Intrusion detection · CN2 · Feature reduction · Classification

✉ Neeraj Kumar
phdcs100009.16@bitmesra.ac.in

Upendra Kumar
upendrakr@bitmesra.ac.in

[1] Computer Science & Engineering Department, Birla Institute of Technology, Mesra, Ranchi (Patna Campus), Patna, Bihar, India

1 Introduction

In the era of internet and with the advent of new technologies network security is one of the most important concerns. Network Intrusion Detection Systems (NIDS) are the solutions for the security of these networks to keep monitoring over the network traffic and alarm when an intrusion has occurred. NIDS can be broadly classified into two categories: misuse detection system and anomaly detection system. The misuse detection system is designed to identify intrusions that match known attack scenarios while the anomaly detection system attempts to detect intrusion by searching for the malicious behaviour that deviates from the established normal patterns. Researchers focused to develop an intrusion detection system (IDS) model which is more intelligent and reliable and it has been observed that the performance of the NIDS model is associated with classification algorithms and training datasets. So, choosing an intelligent classifier is also a major concern that influences the performance of any predictive model like NIDS [10, 16].

Indeed, NIDS is designed for the detection of various attacks by examining lots of previous data records observed during the processes on the network. However, in the case of big data major challenges were to attain a maximum rate of detection accuracy and reducing the computational complexity. Perhaps, use of an intelligent classifier to distinguish between attack and normal traffic packets over the network and attained maximum detection accuracy with the low computational cost is of paramount importance. It is because everybody is not statistically proficient in analyzing the data properties and selects a suitable classifier. Therefore, most of the popular methods entail sampling of train-set and testing over the various classification algorithms and subsequently compare the performance on test-set to select the classifier with the highest performance. In the early stages, rule-based [22] and statistical methods [28] were focussed. But, when the larger datasets are encountered, especially the datasets with a large number of dimensions, rule-based classifiers become less effective. So, large numbers of techniques using Data Mining (DM) and Machine Learning (ML) have been used to overcome the problem.

A rule-based system is an expert system model that contains a set of rules usually in the form of If-then. These rules are framed with the trends analysis and with the help of expert knowledge learning from the real world. Indeed, rule-based system models are more reliable in many tasks such as classification, regression, association, and prediction. In [15], two rule learning approach known as ‘Divide and Conquer (DAC) and ‘Separate and Conquer’ (SAC) are discussed. Moreover, Top-Down Decision Tree (DT) approach is also implemented for this purpose [27]. It is pertinent to mention that there is a subtle difference between DT learning and rule learning. In DT learning, attributes selection takes place right from root to leaf node whereas the functioning of rule learning is based on attribute-value pairs. Moreover, in DT approach usually, the production of complex rule in large numbers is formed due to replicated sub-tree which leads the problem of maximizing the computational cost and overfitting of training data. Further, some problems are also associated with rule-based DAC-ML techniques such as most of the algorithms require heavy computational resources and are prone to underfitting or overfitting. Even more, we can’t change the decisions or rules of these classifiers at any time if we are not convinced by the results [14]. Furthermore, rule Induction classifiers based on SAC can be used to overcome these shortcomings. Rule-based induction algorithms don’t prove efficient for larger datasets with huge dimensions. However, optimization techniques such as reducing the dimensionality of the dataset can help these algorithms to achieve efficacy to deal even with larger datasets.

It has been observed from the literature review that not as much work related to rule-based techniques has been carried out as with other DM and ML techniques. Perhaps, the reason behind this was the inefficiency of the rule-based classifiers to handle the large datasets or the datasets with a large number of dimensions. Moreover, it has also been revealed that even the DM and ML algorithms are not flawless. Most of them require many computational resources and can fall victim to overfitting or underfitting problems. Following are few challenges observed during the survey:

- The problem of underfitting and overfitting.
- High computational cost for predictive model.
- Rule replication problem wherein sub-trees can be repeated on different branches in the case of DT.
- Low classification accuracy in case of SAC rule-based system.
- Rules are not allowed to customize manually at any point as desired.

In this paper, we have proposed a technique which reflect new paradigm of CN2 rule induction algorithm for the NIDS model based on the SAC strategy. In this proposed technique initially, the pre-processing of the dataset is done. Pre-processing of the dataset is executed basically in two phases. In the first phase, redundancy is removed from the dataset by removing the duplicate entries. In the second phase, the dimensions of the dataset are removed by choosing a subset of features available in the dataset. Further, the reduced dataset is fed to the classifier. The experiment is performed using a Python programming language-based software called 'Orange' and a subset of features of the Kddcup99 dataset is used as a test-bed. The objective of this work is to analyze the performance of different methods and to comprehend how they can be fitted to the NIDS model. The main contributions of the proposed method are enumerated below:

- The functionality of proposed technique is simple.
- The pre-processing of the dataset is performed effectively.
- The proposed method can perform efficiently even if there is redundancy in the dataset.
- The proposed method can also efficiently be used for the high-dimensional dataset.
- The proposed technique can easily be ensemble with different classifiers.
- The proposed technique reduces the problem of overfitting and underfitting.
- The time consumed in training is low.
- The proposed method is compared with different classifiers like K-Nearest Neighbour (KNN) [29], Random Forest (RF) [19], DT [3], Naïve Bayes [17] and Multi-Level Perceptron (MLP) [9].
- Different standard evaluation metrics like the ROC curve, the area under the curve (AUC), classification accuracy (CA), F1-Score, Precision, and Recall [6] are used for comparative analysis.

The rest of the paper is organized as follows. In Section 2, the related works on the IDS have been presented. Section 3 describes the framework of the proposed method. Further, Section 4 renders the inductive learning process and CN2 algorithms. Section 5, depicts the experimental work about the proposed model as well as analysis and discussion of the results. Finally, the paper is concluded in Section 6.

2 NIDS learning related work

The plethora of research work about NIDS is revealed during the literature survey. In [18], a comparative study of unsupervised learning algorithms is presented to detect the anomaly type of unknown attack. Further in [18], a combinatory approach using supervised and unsupervised is also applied to the NSL-KDD dataset and ISCX dataset to determine the anomaly type of attack. Moreover, a comparative study has been carried out using Isolation Forest, Auto-encoder, Nearest Neighbour (NN), K-Means, Scaled Convex Hull (SCH) and One-Class Support Vector Machine (OC-SVM) for investigation. The One class Nearest Neighbour (1-NN) algorithm has also been used to detect zero-day attacks and they applied the Singular Value Decomposition (SVD) approach for dimension reduction [18]. In [26], a study is performed on attack classification using feature selection (FS) and emphasis is given on removal of features that do not participate in the class prediction of an attack. Further, chi-square, Information-gain and recursive feature elimination techniques are used for feature selection, and subsequently, these selected features are used in different ML-based classifiers such as support vector machine (SVM), DT, Naïve Bayes, RF, Logistic Regression and Neural Networks (NN). Moreover, the comparative results of different classifiers onto the NSL-KDD dataset are analyzed in [26]. Further, in [21], research work on land mapping and forest data visualization are given in which comparative analysis has been executed on various rule-based classifiers like SVM, Logistic Regression, KNN, DT, Stochastic Gradient Descendent, and CN2 Rule Induction. In [21], the performance of these classifiers is also measured in terms of accuracy and AUC. Furthermore, in [15], rule-based systems and granular computing as well as the relationships in the context of computational intelligence using divide and conquer are introduced. In [15], it has been suggested that fuzzy and rough logic can be used for handling uncertainty effectively through a reduction in reliability. However, in [15], it is also suggested that the separate and conquer approach-based rule learning algorithms can be used for further enhancement of accuracy. Moreover, in [23], the development of a predictive model has been discussed and it is suggested that the application of a sequential covering rule induction algorithm is the most appropriate solution while developing a prediction system with the comprehensible data model. The research work given in [23] also discuss the performance of separate and conquer based rule learning model in rule generation, classification and regression. Further, in [25], a supervised machine learning system is developed to identify the network traffic if it is malicious or benign. The method given in [25], integrates the supervised learning algorithm and feature selection method. In [25], it is mentioned that the Artificial Neural Network (ANN) based machine learning with wrapper feature selection can outperform the support vector machine (SVM) technique while classifying the network traffic. This method also uses the NSL-KDD dataset for the classification of network traffic using SVM and ANN supervised ML methods. In [4], a designing approach is proposed for NIDS that is based on ML for software-defined networks. The method designed in [4] is also tested for the NSL-KDD dataset by taking only five out of forty-one features. The classification accuracy of this method is 95.95% for identifying the type of attack (DDoS, PROBE, R2L and U2R).

3 Proposed rule-based NIDS model

In this paper, a SAC rule-based model for NIDS is proposed to keep track of network traffic and to ensure the network securities. The proposed model consists of three phases. These are:

- Pre-processing.
- Training.
- Prediction.

In the pre-processing phase, the dataset is retrieved and pre-processing of the dataset is done. In this phase different phenomena such as removal of duplicates, scaling of data, dimensionality reduction, etc. using various techniques take place. After pre-processing, the size of the dataset is generally reduced and thus algorithm performs better. Further, in the training phase, the pre-processed dataset is inputted to the CN2 rule induction model that generates or learns rules for classification. These generated rules are stored in rule-base. Moreover, the CN2 rule induction algorithm also evaluates the rules and decides their quality until the stopping criteria

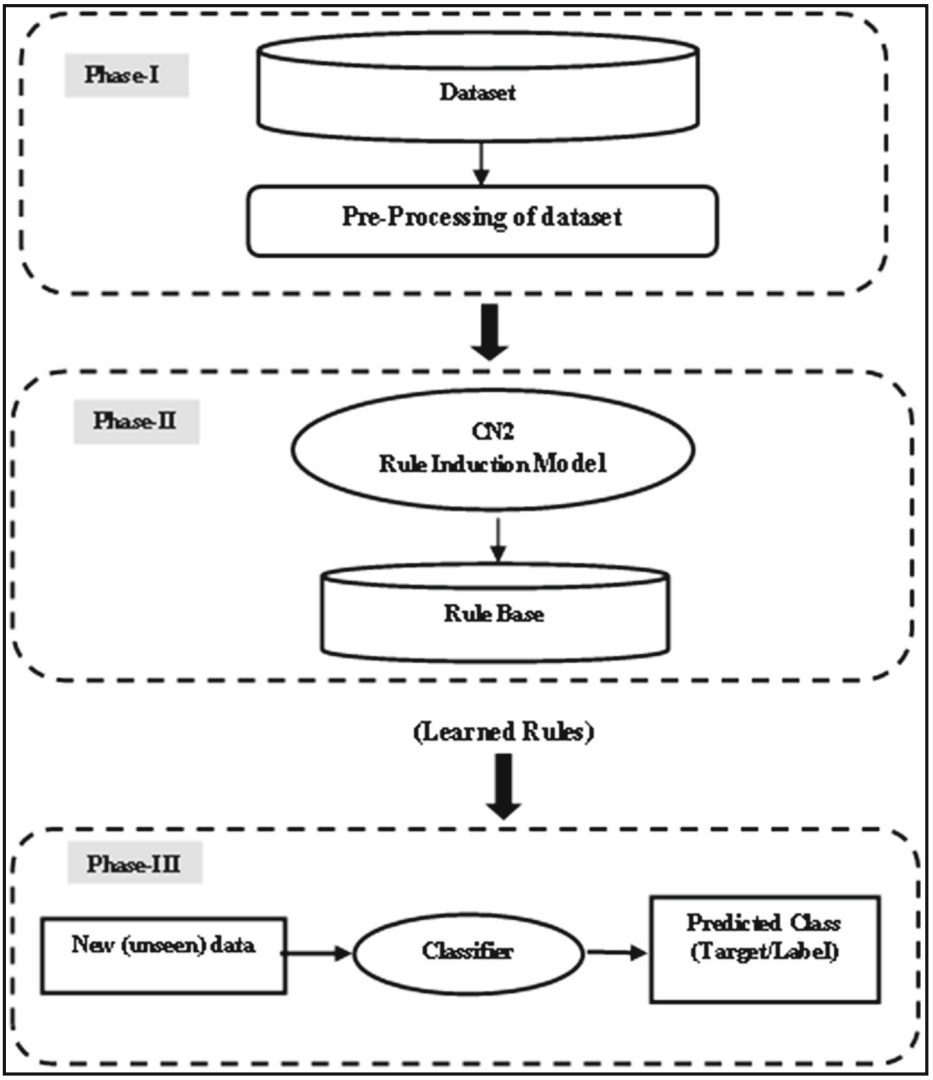

Fig. 1 Framework of proposed method

are reached. Finally, in the prediction phase of the proposed NIDS model, new or unseen data is fed to various classifiers like KNN, RF, DT, Naïve Bayes, MLP and CN2 Rule Inducer for classification and prediction. The workflow of the proposed model is shown in Fig. 1.

4 Inductive learning

Inductive learning is used to identify the training data or earlier knowledge patterns. These knowledge patterns are extracted as generalized rules. As has already been stated in Section 1, inductive learning algorithms are broadly classified into two domains. These are:

- Divide and conquer algorithm (DAC).
- Separate and conquer algorithm (SAC).

Divide-and-conquer algorithms are classification techniques that derive the general conclusions using a decision tree. Some important algorithms from this domain are ID3, C4.5, and CART. However, DAC algorithms have certain merits and demerits like complications arise when some attributes are either repetitive or redundant. In [2], it is mentioned that in this approach it is difficult to handle a tree when it gets too big and thus causing unnecessary confusion. Contrary to this, in separate and conquer algorithms rules are directly induced from a given set of training examples. Some examples of such algorithms are CN2, and algorithms of the family of the RULES extraction system, etc. In general, SAC algorithms give better results in comparison to DAC algorithms. In [23], it is given that direct rules inductions from datasets are advantageous in comparison to rule induction from the decision tree structure.

4.1 The CN2 and CN2 learning algorithms

The CN2 is proposed in [8]. This algorithm entails the essence of Algorithm Quasi (AQ), and the popular decision tree learning algorithm ID3 proposed in [20]. The CN2 algorithm evaluates each possible condition and also recognizes and handles the overfitting problem. Moreover, CN2 includes a pre-pruning technique and thus it filters out all insignificant rules. An advanced version of CN2 is proposed in [7]. In this advanced version, the Laplace measure is used as an evaluation heuristic.

The CN2 learning algorithm follows an iterative way where an attempt is made to search a complex rule that tries to cover a large number of samples of a single class C and a few of other classes. The evaluation function of the CN2 learning algorithm determines whether the complex rule is both predictable and reliable or not. Further, if the complex rule is predictable and reliable then it is included in the rules list. A beam search of the space of complexes is carried out to examine the specializations of only this set. Furthermore, for specializing a complex either a new conjunctive term is added or a disjunctive element is removed from one of its selectors. There are various ways in which each complex can be specialized and all such specializations are generated and evaluated by CN2. CN2 rule induction algorithm is given in Table 1. In [13], it is given that the CN2 rule induction algorithm considers the mid-value of frequently observed sub-range to deal with numeric attributes.

Table 1 CN2 Rule induction algorithm

CN2 Rule Induction Algorithm:
Step 1: Start Step 2: Declare CC → Collection of Classifiers Step 3: Declare QUALIFIED → Collection of qualified selectors Function CN2I (CC) { Step 4: Suppose CONDITION_LIST→ {ϕ} Step 5: do while (BEST_CP=NIL or CC = {ϕ}) Step 6: Let BEST_CP can Search_BEST_COM(CC) Step 7: If (BEST_CP !=NULL) Then suppose CC ' be the set covered by BEST_CP. Compute CC = CC – CC '. Step 8: Declare COM → most common set in CC '. Step 9: Add Rule ' if BEST_CP then the set is COM' at last of CONDITION_LIST. Step 10: Return CONDITION_LIST. Function Search_BEST_COMP(CC) Step 11: Declare ECOMPLEX → Set of Empty Complex i.e. {ϕ} Step 12: Compute BEST_CP = NIL Step 13: While (ECOMPLEX != {ϕ}) Train all complexes in ECOMPLEX by following way: Let NEWECOMPLEX → { K A Z|K 6 ECOMPLEX, Z ∈ QUALIFIED } Compute NEWECOMPLEX = NEWECOMPLEX – ECOMPLEX For all ECOMPLEX = NEWECOMPLEX A ECOMPLEX Or ECOMPLEX = {ϕ} Step 14: For all complex COMi in NEWECOMPLEX : If COMi is statistically significant and better than BEST_CP according to user given conditions when tested on CC, Then BEST_CP = COMi Step 15: Do while size of NEWECOMPLEX is < user specified maximum: Subtract the worst complex from NEWECOMPLEX. Step 16: Compute ECOMPLEX = NEWECOMPLEX. Step 17: Return BEST_CP. Step 18: Stop

5 Experiments and results

In this section, the authors present the entire simulation and experimental work and corresponding results as obtained from different classifiers as well as from CN2 rule induction algorithm. Performance of various classifiers is evaluated using various metric parameters such as Classification Accuracy, Precision, Recall, F1-Score, minimum error with the help of confusion matrix, AUC and ROC, etc. The experiment has been carried out using Python programming language-based software called 'Orange (v3.14)' [11]. After pre-processing on Kddcup99 'corrected-dataset', a subset of the dataset with selected five features is used for processing like rule-generation, training, and testing.

5.1 Dataset description

Kddcup99 dataset is used as benchmark dataset which is prepared and managed by MIT Lincoln Labs. The archive of the dataset consists of three datasets namely: 'corrected-dataset', '10%-dataset', and 'full dataset. This dataset consists of millions of records and each record

contain forty-one features out of which seven are discrete and the rest thirty-four features are continuous. Corresponding to each record a class has been mentioned to which that record belongs. There are thirty-eight classes in the 'corrected-dataset'. These classes are again grouped into six categories namely: DoS, PROBE, R2L, U2R, OTHER, and NORMAL.

In this experimentation, the authors have used the 'corrected-dataset' because it contains all the types of attacks. Further, this dataset originally contained 311,029 records.

5.2 Pre-processing of the dataset

Two actions are performed on the dataset before inputting it to the CN2 Rule Induction and other classifiers. These are:

- Removal of duplicate entries from the dataset.
- Feature reduction.

Initially, this dataset contains 311029 records and after removing the duplicate entries from the dataset, it ended with 77291 records. The distribution of the reduced dataset is shown in Table 2. It is explicit from Table 2 that for PROBE, R2L, and particularly for U2R the sample of the training dataset is substantially less in number. Moreover, out of forty-one features only five features are chosen as a benchmark. The chosen features entail both continuous and discrete features. Continuous features include duration, diff_srv_rate, dst_host_rerror_rate and discrete features include diff_srv_rate and dst_host_rerror_rate. Further, the subset of the dataset is applied to every classification algorithm to maintain uniformity and to justify the study.

5.3 Experimental results

CN2 Rule Induction algorithm-generated 297 rules by considering the following parameters. Rule ordering: ordered, Covering algorithm: exclusive, Gamma: 0.7, Evaluation measure: entropy, Beamwidth: 5, Minimum rule coverage: 1, Maximum rule length: 5, Default alpha: 1.0, Parent alpha: 1.0. Table 3 displays some of these rules for the target class (Normal, DoS, Probe, Others, R2L, and U2R) with their distribution and percentage of probability. It is explicit from the rules given in Table 3 that these rules are denoted as an expression: IF cause (antecedent) THEN effect (consequent). If the condition rendered in the antecedent (Input) part on the left-hand side will satisfy then the 'consequent' part of the rule will be executed and predicted as output. Further, different conjunctive and disjunctive connectives can also be used in the antecedent part. In this experimentation, a total of 297 rules are obtained through CN2 rule induction by using the 'Orange' software for 'corrected-dataset' of Kddcup99 as shown in Table 3. Different pertinent information such as distribution, probability, quality and length of

Table 2 Distribution of the reduced dataset

Connection type	Total no. of occurrence	% of dataset
DOS	21,720	28.10%
NORMAL	47,913	61.99%
PROBE	2682	3.47%
R2L	2328	3.011%
U2R	39	0.050%

Table 3 Generated or Learnt rules viewed in Rule Viewer of Orange

	IF conditions		THEN class	Distribution	Probabilities [%]	Quality	Length
0	flag=OTH	→	label=PROBE	[0, 0, 0, 4,0, 0]	10 : 10 : 10 :50 : 10 : 10	-0.00	1
1	flag=SH	→	label=PROBE	[0, 0, 0, 80,0, 0]	1 : 1 : 1 : 94 :1 : 1	-0.00	1
2	dst_host_rerror_rate≥1.0 AND flag=RSTR	→	label=PROBE	[0, 0, 0, 4,0, 0]	10 : 10 : 10 :50 : 10 : 10	-0.00	2
3	dst_host_rerror_rate≥1.0 AND diff_srv_rate≥0.97	→	label=PROBE	[0, 0, 0, 60,0, 0]	2 : 2 : 2 : 92 :2 : 2	-0.00	2
4	dst_host_rerror_rate≥1.0 AND diff_srv_rate≥0.05	→	label=DOS	[12,574, 0, 0,0, 0, 0]	100 : 0 : 0 : 0: 0 : 0	-0.00	2
5	flag=S3 AND diff_srv_rate≥1.0	→	label=OTHER	[0, 0, 6, 0,0, 0]	8 : 8 : 58 : 8 :8 : 8	-0.00	2
6	duration≥8579.0 AND protocol_type≠tcp	→	label=NORMAL	[0, 3, 0, 0,0, 0]	11 : 44 : 11 :11 : 11 : 11	-0.00	2
7	duration≥7212.0 AND diff_srv_rate≥0.5	→	label=OTHER	[0, 0, 7, 0,0, 0]	8 : 8 : 62 : 8 :8 : 8	-0.00	2
8	duration≥8579.0 AND flag=RSTR	→	label=NORMAL	[0, 3, 0, 0,0, 0]	11 : 44 : 11 :11 : 11 : 11	-0.00	2
9	duration≥7212.0 AND flag=RSTR	→	label=OTHER	[0, 0, 3, 0,0, 0]	11 : 11 : 44 :11 : 11 : 11	-0.00	2
10	duration≥9645.0 AND flag=S2	→	label=NORMAL	[0, 2, 0, 0,0, 0]	12 : 38 : 12 :12 : 12 : 12	-0.00	2
197	duration>=22.0 AND dst_host_rerror_rate<=0.07 AND duration<=73.0 AND dst_host_rerror_rate>=0.07	→	label=U2R	[0,0,0,0,0,1]	14:14:14:14:14:29	-0.00	4
202	duration>=22.0 AND dst_host_rerror_rate<=0.68 AND duration>=684.0	→	label=U2R	[0,0,0,0,0,2]	12:12:12:12:12:38	-0.00	3
204	duration>=22.0 AND duration<=71.0 AND dst_host_rerror_rate>=0.39	→	label=U2R	[0,0,0,0,0,1]	14:14:14:14:14:29	-0.00	3
268	flag==RSTR AND dst_host_rerror_rate<=0.1	→	label=DOS	[36,0,2,2,1,8]	79:2:6:6:4:2	-0.721	2
275	IF protocol_type==icmp AND dst_host_rerror_rate>=0.01	→	label=DOS	[20,16,0,0,1,0]	49:40:2:2:5:2	-1.144	2
286	dst_host_rerror_rate>=0.05	→	label=DOS	[191,95,13,0,1,2]	62:31:5:0:1:1	-1.213	1
287	dst_host_rerror_rate>=0.03 AND duration>=1.0 AND duration>=2.0 AND duration>=3.0	→	label=R2L	[0,1,0,0,2,0]	11:22:11:11:33:11	-0.918	4
288	dst_host_rerror_rate>=0.03 AND duration>=1.0 AND dst_host_rerror_rate>=0.04	→	label=OTHER	[0,11,18,0,0,0]	3:34:54:3:3:3	-0.958	3
289	dst_host_rerror_rate>=0.03 AND dst_host_rerror_rate>=0.04	→	label=NORMAL	[25,53,1,0,0,0]	31:64:2:1:1:1	-0.991	2
290	dst_host_rerror_rate>=0.03 AND duration>=1.0	→	label=OTHER	[0,18,20,1,0,0]	2:42:47:4:2:2	-1.144	2
291	dst_host_rerror_rate>=0.03	→	label=NORMAL	[23,64,0,0,0,0]	26:70:1:1:1:1	-0.833	1
292	duration>=1.0	→	label=NORMAL	[1,52,3,0,14,1]	3:69:5:1:19:3	-1.157	1
293	protocol_type==icmp	→	label=DOS	[911,201,3,256,0,0]	66:15:0:19:0:0	-1.269	1
294	flag!=REJ	→	label=R2L	[27,145,1,0,215,3]	7:37:1:0:54:1	-1.347	1
295	protocol_type==tcp	→	label=OTHER	[0,7,8,3,0,0]	4:33:38:17:4:4	-1.481	1
296	TRUE	→	label=NORMAL	[21720,47913,2609,2686,2328,39]	28:62:3:3:3:0	-1.43	0

Table 4 Performance comparison of various classifiers on various methods (Target class: DoS)

Model	AUC	CA	F1	Precision	Recall
KNN	0.995	0.989	0.980	0.973	0.988
Random Forest	0.998	0.989	0.981	0.974	0.989
Decision Tree	0.992	0.988	0.979	0.978	0.981
Naïve Bayes	0.995	0.976	0.956	0.985	0.929
MLP	0.998	0.988	0.979	0.975	0.9843
CN2 Rule Inducer	0.997	0.989	0.981	0.973	0.989

Table 5 Performance comparison of various classifiers on various methods (Target class: NORMAL)

Model	AUC	CA	F1	Precision	Recall
KNN	0.982	0.964	0.971	0.965	0.977
Random Forest	0.987	0.969	0.975	0.968	0.982
Decision Tree	0.959	0.962	0.970	0.951	0.989
Naïve Bayes	0.970	0.944	0.956	0.929	0.986
MLP	0.987	0.963	0.970	0.961	0.979
CN2 Rule Inducer	0.986	0.969	0.975	0.968	0.982

each rule is also shown in Table 3. Further, these rules are utilized to predict the types of attacks.

Finally, the learned rules are passed to the Test and Score viewer to obtain the Accuracy, Precision, Recall, F-Score, and other evaluation metrics. These results are shown in Tables 4, 5, 6, 7, 8 and 9.

5.4 Evaluation criteria

The authors compared the performance of all the classifiers with CN2 rule induction based on accuracy, sensitivity, specificity, precision, recall and ROC Curve. The values of accuracy, sensitivity and specificity are estimated by True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) as obtained from the confusion matrix. Generally, the measurement criteria used to evaluate the detection precision of the NIDS model are as below:

- True Positives (equivalent. with hit) are used to indicate that the IDS detected an occurred attack precisely.
- True Negatives (equivalent with correct rejection) are used to indicate that the IDS did not make mistake in detecting a normal condition.

Table 6 Performance comparison of various classifiers on various methods (Target class: PROBE)

Model	AUC	CA	F1	Precision	Recall
KNN	0.970	0.990	0.864	0.892	0.838
Random Forest	0.992	0.991	0.873	0.896	0.851
Decision Tree	0.942	0.988	0.822	0.918	0.744
Naïve Bayes	0.981	0.980	0.707	0.742	0.676
MLP	0.996	0.991	0.869	0.888	0.850
CN2 Rule Inducer	0.996	0.991	0.870	0.899	0.844

Table 7 Performance comparison of various classifiers on various methods (Target class: R2L)

Model	AUC	CA	F1	Precision	Recall
KNN	0.886	0.987	0.767	0.873	0.684
Random Forest	0.940	0.987	0.786	0.841	0.738
Decision Tree	0.891	0.987	0.759	0.914	0.649
Naïve Bayes	0.930	0.977	0.477	0.816	0.337
MLP	0.934	0.977	0.597	0.652	0.551
CN2 Rule Inducer	0.921	0.987	0.775	0.821	0.734

Table 8 Performance comparison of various classifiers on various methods (Target class: U2R)

Model	AUC	CA	F1	Precision	Recall
KNN	0.555	0.999	0.000	0.000	0.000
Random Forest	0.723	0.999	0.071	0.125	0.050
Decision Tree	0.540	0.999	0.100	0.250	0.062
Naïve Bayes	0.941	0.999	0.000	0.000	0.000
MLP	0.952	0.999	0.071	0.750	0.037
CN2 Rule Inducer	0.926	0.999	0.061	0.158	0.037

- False Positives (equivalent with False Alarm, Type I error) are used to indicate that a particular attack has been detected but such an attack has not occurred.
- False Negatives (equivalent to miss, Type II error) are used to indicate that the IDS is unable to detect the intrusion while a particular attack has occurred.

But, in the training and test set, the number of instances for U2R, Probe and R2L are too low. Therefore, these measurement criteria wouldn't be sufficient as a standard performance measure. Perhaps, it would result in a bias if these measurement criteria are used for measuring the performance of the system. Keeping this point in consideration, Precision, Recall and F-value/F-score/F-1 score have been used to measure the performance because these are independent of the size of training and testing samples.

The values of Precision, Recall, F1-Score and classification accuracy are obtained from the result set of Orange using Eqs. (1) to (4) as given below.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

Table 9 Comparative evaluation of the overall performance of the data mining classifier algorithms using Orange software

Model	AUC	Classification Accuracy (CA)	Training Time in Second	F1	Precision	Recall
KNN	0.908	0.958	67.32	0.957	0.956	0.958
Random Forest	0.952	0.963	484	0.962	0.962	0.963
Decision Tree	0.874	0.956	118	0.954	0.956	0.956
Naïve Bayes	0.966	0.929	5.67	0.924	0.926	0.929
MLP	0.979	0.953	349.98	0.951	0.951	0.953
CN2 Rule Inducer	0.976	0.962	6.57	0.961	0.961	0.962

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{F} - \text{value} = \frac{2 * \text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})} \tag{3}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{4}$$

5.5 Results analysis

In this experimentation, the authors fed the same pre-processed dataset to the CN2 rule inducer and five other important algorithms of DM and ML for the task of classification. Other five classifiers implemented are KNN, Random Forest, Decision Tree, Naïve Bayes, and MLP. Performance of all the classifiers is tabulated by considering different sampling types such as Stratified Shuffle split and 10 random samples with 80% data on 'Orange'. Further, ROC curves for DOS, NORMAL, OTHERS, PROBE, R2L and U2R are drawn as shown in Figs. 2, 3, 4, 5, 6 and 7 respectively with random predictor threshold AUROC of 0.5. From this experimentation, it has been observed that the performance of the CN2 rule inducer is considerably superior to some of the most popular classifiers of ML and DM. It is explicit from Tables 4, 5, 6, 7, 8 and 9 that the CN2 rule inducer has appreciably considered all the classes even though the numbers of samples are very low as in the case of R2L and U2R. Contrary to this, some of the selected classifiers have gone miserable to detect those classes. However, CN2 handled it gracefully (of course the precision has gone low but at least it has

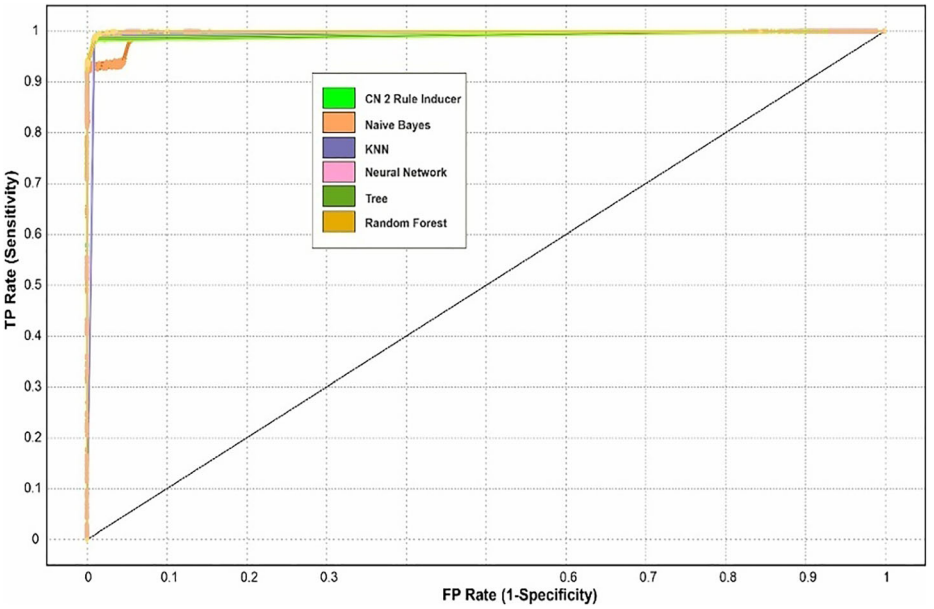

Fig. 2 ROC for DoS

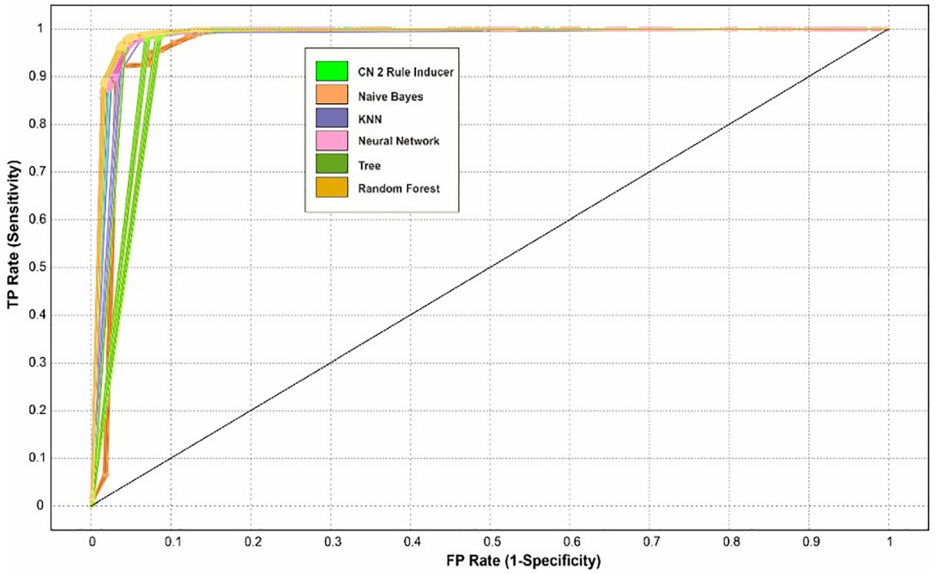

Fig. 3 ROC for NORMAL

not gone zero or very poor). From the experimental results, as shown in Tables 4, 5, 6, 7, 8 and 9, it appears that MLP has been found the victim of overfitting as its precision has gone 100 percent on very few samples. Further, from Recall and F-value as shown in Tables 4, 5, 6, 7, 8 and 9, it is quite evident that CN2 has performed at par with RF and performed well in comparison to the rest of the classifiers. Furthermore, from the ROC curve shown in Figs. 2, 3, 4, 5, 6 and 7, it has been observed that AUC for CN2 rule induction classifier is above 92.1% for the classification of all attack classes and consistently performed well as compared to most

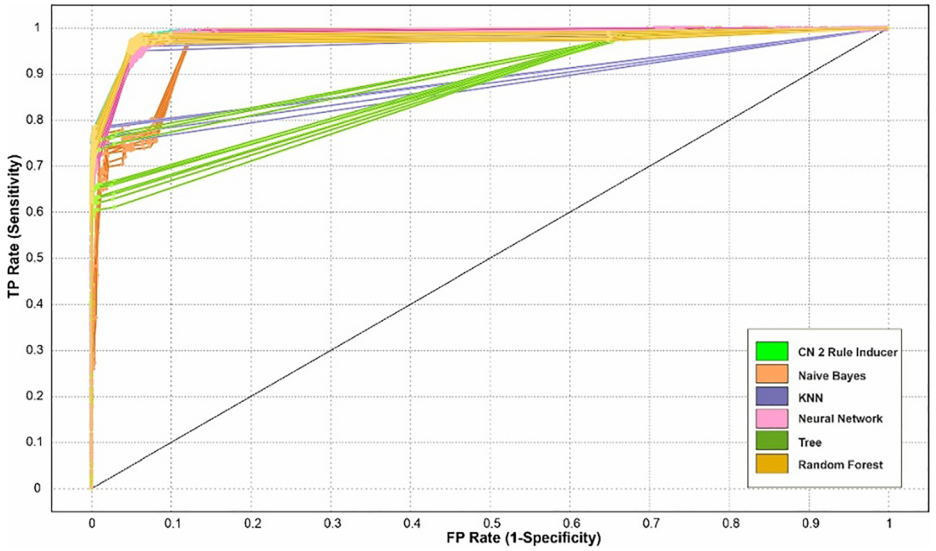

Fig. 4 ROC for Others

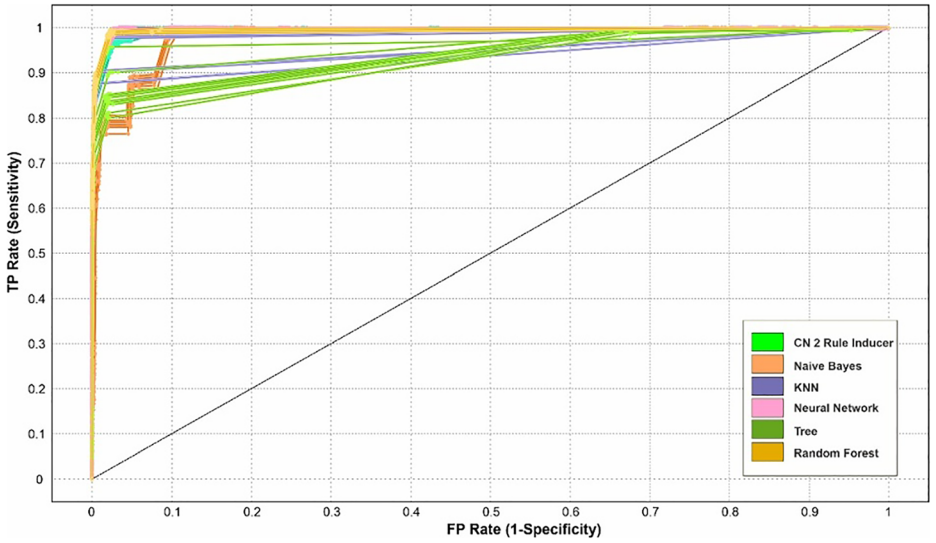

Fig. 5 ROC for PROBE

other classifiers. It is also obvious from Fig. 2 and Table 4 that the performance of the CN2 rule induction classifier is at par with other classifiers for the 'DoS' class. Moreover, Fig. 3 and Table 5 render the fact that the classification accuracy of the CN2 rule induction classifier and the random forest is very near to one, unlike other classifiers. Even more, for the 'PROBE' class type the classification accuracy of CN2 rule inducer, RF, and MLP is 99.1% as shown in Table 6. Indeed, in this experimentation, CN2 performed well even when the sample size of training and test set is very less like R2L and U2R. The classification accuracy of the CN2 rule inducer is compatible with other ML techniques for the R2L class type as shown in Table 7. However, the classification performance of all the classifiers is the same i.e. 99.9% for class type 'U2R' as shown in Table 8. From Table 9, it is explicit that the AUC-ROC for CN2 rule

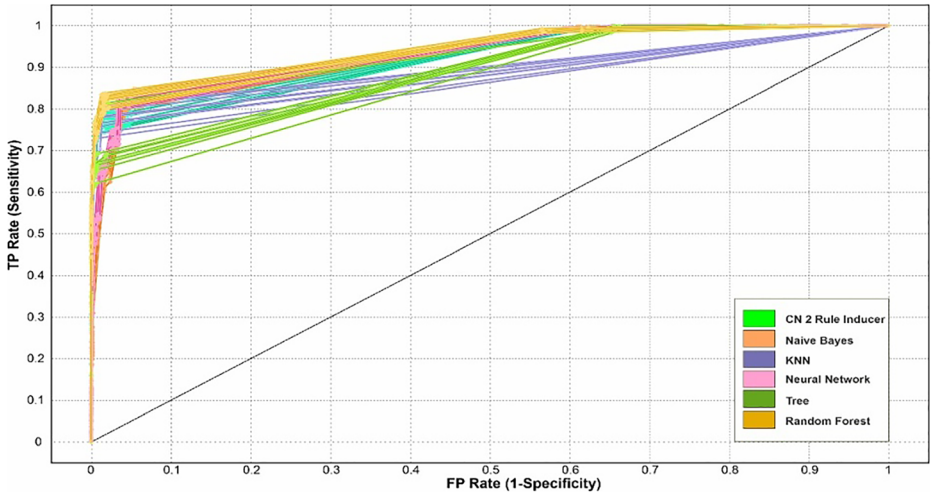

Fig. 6 ROC for R2L

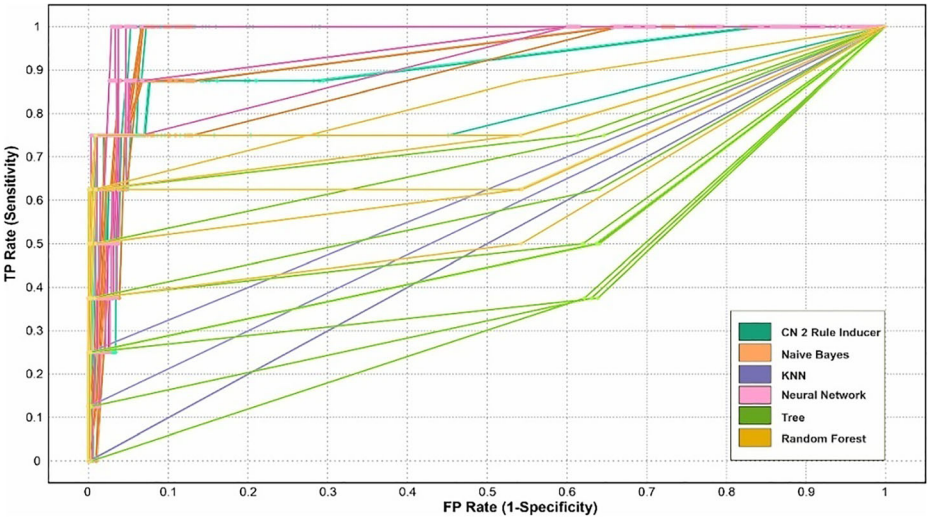

Fig. 7 ROC for U2R

Table 10 Comparison of the average classification accuracy of proposed method with some latest works

Author	Dataset	Average Classification Accuracy (CA) %
Our method	KDD CUP 99	96.20
Akashdeep [1]	KDD CUP 99	94.31
Sun [24]	KDD CUP 99	56.80
Jia [12]	KDD CUP 99	94.24
Andresini [5]	KDD CUP 99	92.49

inducer is 0.976 that reveals the fact that there are 97.6% chances to distinguish between positive and negative classes. Further, Table 9 also displays the fact that the performance of the CN2 rule inducer is satisfactory and it performs better than the classifiers considered in this experimentation except for the RF. However, its classification accuracy is also very near to that of RF. There is only a subtle difference of 0.1%. It is also explicit from Table 9 that the training time of CN2 rule inducer is only 6.57 second that is slightly greater than the training time of Naive-Bayes classifier whose training time is 5.67 second but is substantially better than the other classifier.

Further, authors also compared the average classification accuracy of proposed CN2 rule induction algorithm with some latest work as given in [1, 5, 12, 24]. These results are displayed in Table 10. It is explicit from Table 10 that performance of CN2 rule induction algorithm is substantially superior.

6 Conclusions and future directions

In this paper, the authors proposed and implemented SAC based technique using CN2 rule induction algorithm and compared it with some state-of-art ML techniques for NIDS and validated it using the Kddcup99 dataset. A subset of features of the Kddcup99 dataset is selected and subsequently, the CN2 rule induction classifier is used to perform the classification task. The

experimental results render the fact that the performance of proposed algorithm is almost similar to RF and performs better in comparison to many other ML algorithms. It is also explicit from experimental results that the proposed algorithm performs well even with such a large dataset and is also advantageous for detecting the samples with low frequency for which some of the classifiers fail or perform miserably. Moreover, the proposed method also reduces the problem of underfitting and overfitting. The overall AUC and classification accuracy of the proposed method is 97.6% and 96.2% respectively that implicates its reliability over the other algorithms. Its computational cost is also low as compared to other ML and DT-based rule learning algorithms. The performance of proposed method is also superior to many latest published works.

At present, network intrusion detection is not a fully developed technology and thus there is a large scope of future research as its application domains are substantially diversified. Moreover, the risk of having a greater potential attack would not be rejected with conformity forever, so the scope of future research enhancement will always be associated with this field. In the future, further enhancements can be done to improve the efficiency of the system by harnessing the potentials of 'SAC' algorithms.

Acknowledgements Authors are grateful to Dr. Subhash Chandra Pandey for his consistent support and valuable suggestions for reviewing of the manuscript. Authors are also highly indebted to learned anonymous reviewers whose valuable comments help us a lot to enhance the quality of this paper.

References

1. Akashdeep IM, Manzoor I, Kumar N (2017) A feature reduced intrusion detection system using ANN classifier. Expert Syst Appl 88:249–257
2. Almana AM, Aksoy M (2014) An overview of inductive learning algorithms. Int J Comput Appl 88(4):20–28
3. Alqahtani H, Sarker IH, Kalim A, Hossain SMM, Ikhlaq S, Hossain S (2020) Cyber intrusion detection using machine learning classification techniques. In: International conference on computing science, communication and security. Springer, Singapore, pp 121–131
4. Alzahrani AO, Alenazi MJF (2021) Designing a network intrusion detection system based on machine learning for software-defined networks. Future Internet 13:111. https://doi.org/10.3390/fi13050111
5. Andresini G, Appice A, Mauro ND, Loglisci C, Malerba D (2020) Multi-channel deep feature learning for intrusion detection. IEEE Access 8:53346–53359
6. Basori AH, Malebary SJ (2020) Deep reinforcement learning for adaptive cyber defense and attacker's pattern identification. In: Advances in cyber security analytics and decision systems. Springer, Cham, pp 15–25
7. Clark P, Boswell R (1991) Rule induction with CN2: some recent improvements. In: European working session on learning. Springer, Berlin, pp 151–163
8. Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3(4):261–283
9. Ghanem WA, Jantan A (2019) A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm. Neural Comput Appl:1–34
10. Hakim L, Fatma R (2019) Influence analysis of feature selection to network intrusion detection system performance using NSL-KDD dataset. In: 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE). IEEE, pp 217–220
11. Hosseini S, Sardo SR (2020) Data mining tools-a case study for network intrusion detection. Multimed Tools Appl: 1–21
12. Jia Y, Wang M, Wang Y (2019) Network intrusion detection algorithm based on deep neural network. IET Information Security 13(1):48–53
13. Knutas A, Van Roy R, Hynninen T, Granato M, Kasurinen J, Ikonen J (2019) A process for designing algorithm-based personalized gamification. Multimed Tools Appl 78(10):13593–13612

14. Liu H, Cocea M (2019) The granular computing-based approach of rule learning for binary classification. Granul Comput 4(2):275–283
15. Liu H, Gegov A, Cocea M (2016) Rule-based systems: a granular computing perspective. Granul Comput 1(4):259–274
16. Martinez MAQ, Rugel DTL, Alcivar CJE, Vazquez MYL (2020) A framework for selecting classification models in the intruder detection system using TOPSIS. In: International conference on human interaction and emerging technologies. Springer, Cham, pp 173–179
17. Meerja AJ, Ashu A, Kanth AR (2019) Gaussian naïve bayes based intrusion detection system. In: International conference on soft computing and pattern recognition. Springer, Cham, pp 150–156
18. Meira J et al (2019) Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. J Ambient Intell Humaniz Comput:1–13
19. Nanda NB, Parikh A (2019) A hybrid approach for network intrusion detection system using random forest classifier and rough set theory for rules generation. In: International conference on advanced informatics for computing research. Springer, Singapore, pp 274–287
20. Quinlan JR (1986) Induction of decision trees. Mach Learn 1(1):81–106
21. Radha Krishnan S et al (2020) Forest data visualization and land mapping using support vector machines and decision trees. Earth Sci Inform:1–19
22. Sangeetha S, Haripriya S, Mohana Priya SG, Vaidehi V, Srinivasan N (2010) Fuzzy rule-base based intrusion detection system on application layer. In: Meghanathan N, Boumerdassi S, Chaki N, Nagamalai D (eds) Recent trends in network security and applications. CNSA 2010. Communications in Computer and Information Science, vol 89. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14478-3_3
23. Sikora M, Wróbel Łukasz, Gudyś A (2019) GuideR: a guided separate-and-conquer rule learning in classification, regression, and survival settings. Knowl Based Syst 173:1–14
24. Sun C, Lv K, Hu CZ et al (2018) A double-layer detection and classification approach for network attacks. In: Proceedings of the 27th International Conference on Computer Communication and Networks (ICCCN), pp 1–8, Hangzhou, China
25. Taher KA, Mohammed Yasin B, Jisan, Rahman MM (2019) Network Intrusion detection using supervised machine learning technique with feature selection, 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), pp 643–646
26. Thakkar A, Lohiya R (2020) Attack classification using feature selection techniques: a comparative study. J Ambient Intell Humaniz Comput:1–18
27. Teli M, Singh R, Kyada M, Mangrulkar R (2020) Network intrusion detection system using machine learning approach. In: Vasudevan H, Michalas A, Shekokar N, Narvekar M (eds) Advanced computing technologies and applicationlogies and Applications. Algorithms for intelligent systems. Springer, Singapore. https://doi.org/10.1007/978-981-15-3242-9_25
28. Waskita AA, Suhartanto H, Persadha PD, Handoko LT (2013) A simple statistical analysis approach for intrusion detection system. IEEE Conference on Systems, Process & Control (ICSPC), 2013, pp 193–197. https://doi.org/10.1109/SPC.2013.6735130
29. Wazirali R (2020) An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation. Arab J Sci Eng:1–15